TRY CATCH EXCEPTION HANDLING

09 February 2025 14:01

```
package dummy;

public class Test {
    public static void main(String[] args) {

        int a = 10, b = 0, c;
        c = a / b;
        System.out.println(c);
    }
}
```

- Whenever there is an exception , the method in which exception occurs will create an object and that object will store three things;

1. Exception Name(Arithmetic exception)
2. Description(/ by zero)
3. Stack Trace(line no print like 7)

- If we do not handle the exception in the code using a try-catch block, the **JVM's Default Exception Handler** will automatically come into the picture. how it works:
  1. **What Happens:**
     When the ArithmeticException is thrown in the program, and there is no mechanism (like a try-catch block) to handle it, the JVM allows the exception to propagate.
  2. **Who Handles It:**
     The **Default Exception Handler** in the JVM takes responsibility for handling unhandled exceptions.
  3. **What It Does:**
     The Default Exception Handler performs the following:
     - Prints the **Exception Name** (ArithmeticException).
     - Displays the **Description** (/ by zero), which explains what caused the exception.
     - Shows the **Stack Trace**, which points to the exact line in the code where the exception occurred (line 7 in this case).

- There are multiple ways to handle the exceptions:

1. Try 2. catch 3. finally 4. throw 5. throws

- Syntax of try-catch:

```
try {
// Code that may throw an exception
// risky code
}
catch (ExceptionType1 e1)
{
// Code to handle ExceptionType1
}
```

- **Methods to print Exception** :

1. printStackTrace()

- Prints the exception name, description, and the complete stack trace (all the methods called before the exception

occurred).

2. getMessage()

- Returns only the description (message) of the exception.

3. toString()

- Returns a string that contains the exception name and its description/message.

Mostly used method is printStackTrace();

- Various possible combination of try catch finally

a. If catch block contains Exception class then I can't create another catch block
b. We can't provide the same Exception class in both catch blocks.
c. We can create nested try catch blocks .
d. We can create nested try catch block inside the catch block also.(reason like data base connnectivity for example one try block contains sql connectivity and inside catch I have another try and catch so in this try block I have oracle database connectivity.)
e. In the finally block I can create try and catch blocks.